COLLOQUE NATIONAL SUR LE TRAITEMENT DU SIGNAL
ET SES APPLICATIONS

NICE du 26 au 30 AVRIL 1977

MACHINES PROGRAMMABLES ADAPTEES AU TRAITEMENT DU SIGNAL

P.LAIDET
Conseiller scientifique auprès de la CIT/ALCATEL

CIT/ALCATEL 1 AVENUE ARISTIDE BRIAND ARCUEIL

RESUME

Dans le cadre du traitement du signal on rencontre deux types de processeurs :

- l'automate réalisant des séquences de calculs périodiques
- le processeur de traitement, qui en plus de la fonction automate, possède des états de fonctionnement (introduits par un opérateur) et éventuellement synchrnose des processus.

Dans le cas de l'automate, l'utilisation de calculateurs universels n'est pas souhaitable car ceux-ci disposent de possibilités hardware non utilisées et exécutent trop lentement les tâches de calcul. Dans le cas du processeur de traitement, la partie automate est mal résolue.

On est donc amené à développer deux types de processeurs correspondant chacun à un type d'utilisation.

L'exposé qui s'appuit sur des études ayant débouché sur une réalisation, propose une structure biprocesseurs utilisant de tels automates adaptée aux problèmes spécifiques liés au traitement du signal.

SUMMARY

Two types of processors may be used in signal processing :

- the automaton, which carries out sequences of periodical calculation
- the computer, which in addition to its automaton, also possesses working statements (introduced by an operator) and can therefore synchronise the processes.

In the case of the automaton, employment of universal calculators is not recommended, as these have hardware facilities which are not used and they perform the calculation operations too slowly. On the other hand, the "treatment" processor is not very effective in executing the automaton operations.

Thus it is necessary to develop two types of processors, each corresponding to a particular use.

The paper, based on studies of the realization of this project, describes a biprocessor structure using automatons adapted to the specific problems of signal processing.

MACHINES PROGRAMMABLES ADAPTEES AU TRAITEMENT DU SIGNAL

LES PROBLEMES RENCONTRES EN TRAITEMENT DU SIGNAL

Lorsqu'on étudie les moyens nécessaires pour traiter les problèmes liés au traitement du signal, on constate que seuls des ensembles modulaires offrent la souplesse d'adaptation, de mise au point et d'évolution, compatibles avec le contexte industriel dans lequel ces matériels doivent être réalisés.

Parmi les ensembles modulaires, ceux qui sont programmables offrent des possibilités d'adaptation très intéressantes. De plus ils peuvent, avec les technologies actuelles, couvrir un éventail assez large des problèmes rencontrés couramment dans le traitement du signal.

Lorsqu'on étudie ces problèmes, on constate un certain nombre de propriétés peu courantes en calcul temps réels :
- fréquence élevée de répétition des tâches
- algorithmes mettant en jeu des fonctions mathématiques assez complexes
- dynamique importante des variables
- recherche d'une grande précision dans certains calculs
- logique d'enchaînement des modules assez simple
- peu de dialogue avec l'extérieur, au moins en ce qui concerne la variété des informations.

CARACTERISATION DES LOGIQUES PROGRAMMABLES ADAPTEES AU TRAITEMENT DU SIGNAL

Si l'on cherche à définir l'ensemble programmable idéal pour résoudre ces problèmes, on rencontre des contraintes qui nécessitent la mise en oeuvre d'éléments très disparates. On trouvera entre autre la faiblesse des échanges et des enchaînements de tâches par rapport au volume de calcul à exécuter.

Il faut donc que l'ensemble programmable idéal possède une (ou des) unité(s) de calcul très performantes en vitesse d'exécution, en dynamique et précision des mots traités.

La vitesse d'exécution est régie par la vitesse intrinsèque de l'unité arithmétique mais également par la richesse du code opération et le nombre de mots accessibles sans référence externe à l'unité arithmétique (registres).

La dynamique nécessite en principe le traitement de valeurs flottantes. Pour les projets traités jusqu'ici la double ou la triple précision a permis d'éviter cette difficulté. Néanmoins les opérations flottantes peuvent en général être simulées dans les unités arithmétiques classiques.

Enfin la précision, souhaitée dans les calculs, peut être obtenue en simulant des valeurs occupant plusieurs mots. Ceci a pour effet de réduire les performances de la machine puisqu'on diminue le nombre des registres et qu'il faut plus d'instructions pour coder une même opération arithmétique. Du fait que ce type de traitement est en général utilisé dans les séquences les plus répétitives, des facilités hardware doivent être offertes au programmeur pour réaliser ces tâches arithmétiques. Parmi ces facilités figurent le traitement des nombres signés et non signés, la notion de sous-programme multiniveaux, le décalage indexé, ...

L'ensemble de ces caractéristiques correspond à celui d'un automate de calcul. Si, pour celui-ci, de bonnes performances sont nécessaires, il ne faut pas être contraint de surveiller les événements extérieurs. Ceci entraîne soit l'adjonction d'un système d'interruption, soit la réalisation de cette tâche extérieurement à l'automate de calcul.

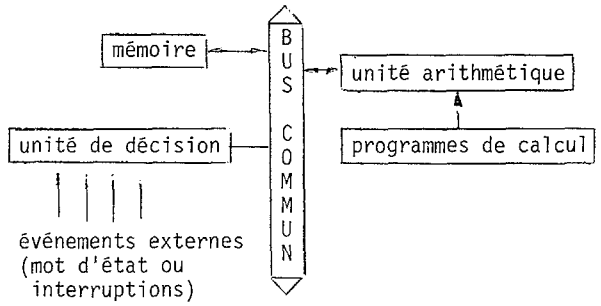
Si le choix se porte sur un système d'interruption, celui-ci doit être puissant. En effet, il faut sauver et initialiser un contexte complexe (nombreux registres, code conditions, etc...) ce qui est pénalisant en temps. De plus dans des applications temps réel ayant une fréquence d'événement élevée, il faut morceler les tâches. Pour cela on est amené à créer et gérer des files d'attentes afin de répartir dans le temps les pointes de charge. Toutes ces tâches nécessitent des possibilités de traitement assez différentes de celles demandées par le calcul arithmétique.

Si l'on veut réunir toutes les possibilités énoncées ci-dessus dans une même unité centrale on obtient de machines non compatibles avec les exigences d'environnement des projets étudiés. Ce mélange possède d'autres défauts, tels que :
- mauvais rendement de l'ensemble. A cause des pointes temps réel on est obligé de ne pas utiliser d'importantes ressources (jusqu'à 50 % de la puissance du processeur)
- complexité due à l'imbrication des fonctions réalisées
- du fait de ces imbrications, les évolutions de spécification peuvent demander un important travail pour être intégrées
- mauvais rendement au niveau des sous-ensembles ayant pour cause l'exclusion des tâches au niveau de l'exécution.

SEPARATION DES FONCTIONS

Une solution aux problèmes énoncés ci-dessus consiste à séparer les fonctions dans des unités différentes. Ainsi l'utilisation d'une unité arithmétique ayant des performances adaptées au problème à résoudre, pilotée par une logique de décision fera disparaître les critiques énoncées contre les systèmes mono-unité.

Dans une telle organisation les deux unités dialoguent par l'intermédiaire d'une mémoire commune. Dans cette mémoire l'unité de décision poste le descripteur de la prochaine tâche qui devra être exécutée par l'unité arithmétique, elle y lit les informations concernant l'exécution de la tâche précédente par l'unité arithmétique.



En regardant ce schéma on constate la mise en évidence d'un bus commun, ce choix fondamental résulte des contraintes technologiques qui obligent à véhiculer l'information entre les organes de stockages et les unités de traitement. Cette voie d'échange existe quelles que soient les organisations retenues. De ses performances dépendent celles du système complet. On cherchera donc à lui donner des performances optimales en particulier on s'efforcera de réduire les temps d'occupation du bus, soit en utilisant de nombreux registres, soit en rendant élémentaires et peu contraignants les accès à ce bus.

Une telle organisation peut fonctionner de la façon suivante : l'unité de décision, mémorise les événements, réalise les acquisitions, crée et gère les files d'attente, définit à l'unité arithmétique la liste des tâches à exécuter. Après traitement par l'unité arithmétique, il y a décision sur les nouveaux événements à activer et sortie des résultats (cadencement avec les organes externes, etc...).

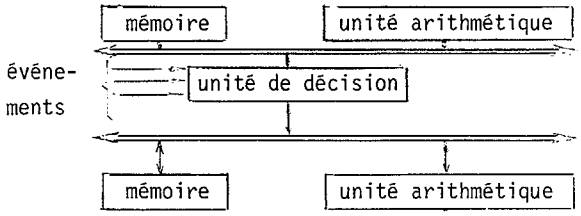
L'unité arithmétique possède toutes les facilités souhaitées pour les monoprocesseurs. Par son mode de travail, sa réalisation est simplifiée car il n'y a plus de registres pour les traitements des interruptions, plus de logique d'échange, plus de découpage temporel des tâches.

Les avantages sont nombreux :

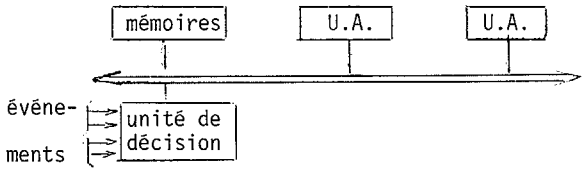
- découpage physique correspondant au découpage fonctionnel et imposant les notions de programmation structurée
- travail en parallèle, donc meilleur rendement des sous-ensembles
- séparation des tâches en deux parties, permettant au programmeur de se dissocier des tâches annexes
- facilité d'évolution pour les tâches de calcul
- le rendement global peut être assez bon car seule la logique de décision nécessite des ressources pour les pointes temps réel.

A partir des principes énoncés ci-dessus on peut envisager :

- des traitements de chaînes parallèles avec une seule unité de décision pilotant plusieurs unités arithmétiques attachées chacune à leur bus



- des traitements séries ou répartis avec une seule unité de décision pilotant plusieurs unités arithmétiques sur un même bus



PERSPECTIVES

L'apport des composants modernes, entre autre les micro-processeurs et les UAL évoluées permettent une réalisation de ces systèmes modulaires et adaptatifs qui sont capables de traiter les charges de calcul compatibles avec certaines applications liées au traitement du signal et ceci sans mettre en oeuvre un matériel trop important.

Parmi toutes les solutions possibles, les unités à code opération adaptatif permettent de réaliser les sous-ensembles nécessaires à ces organisations distribuées. Jusqu'ici ceci ne pouvait être fait qu'avec des machines micro-programmées. L'annonce des micro-processeurs à code adaptatif permettra des réalisations mettant en oeuvre un matériel moins important.

Dans les années à venir nous assisterons à l'abandon des coutumes actuelles qui consistent à adapter la description des algorithmes aux structures des machines et aux habitudes de codage. Dès maintenant le mouvement est amorcé et on commence à voir apparaître des architectures conçues à partir des applications qui devront y être codées.

REFERENCES

A. Structures machines

(1) Symposium on Computer Architecture (1974-1975 et 1976) IEEE

(2) Ivan FLORES Computer Design Prentice Hall 1967

(3) MICROCOMPUTER FORUM Conférence Proceeding 1975

(4) C. GORDON BELL - JOHN GRASON - ALLEN NEWELL Designing computers and digital systems

B. Génération des programmes, structures des données

(5) BARROW D.W. Assemblers and Loaders Mac Donald and Elsevier

(6) Ivan FLORES Assemblers and Bal Prentice Hall 1971

(7) Proceeding of a symposium on data structures in programming languages Université de Floride, Gainesville 22 - 25 février 1971

(8) RUSSEL Robert D A programming language for the Dec PDP - 11 Computer Streater (CERN 1974)

(9) A ADAM et J.P LAURENT Décomposition du graphe d'un programme permettant d'étudier la permutabilité des groupes d'instructions 2ème colloque international sur la Programmation Avril 1976 (Université de Paris 6)

(10) AZRA J.P et JAULIN B. Récursivité Gauthiers Villars 1973

C. Autres éléments de bibliographie

(11) KNUTH D. The Art of computer Programming Addison Wesley

(12) DIJKSTRA E.W. The structure of the "The" Multiprogramming system ACM Vol.11 number 5 May 1968

(13) Papers on digital signal processing Oppenhein Editor

(14) B. FLAVIGNY Detection a priori des erreurs dans les programmes (Thèse 3ème cycle, Paris 6)

(15) JJ. CHEVREUL : R GACHES et J.RACINE Microdiagnostic Congrès Afcet Grenoble 1972.